EXHIBIT B

```
package KnowledgeAgents;

import se.*;
import gui.*;
import utils.*;

import java.awt.*;
import java.awt.event.*;
import java.util.Stack;
import java.io.*;

/**
 * The user interface of the Agent.
 */
public class AgentGUI extends Frame
                      implements ActionListener, ItemListener
{
  Agent agent;
  Trace trace;

  /* private gui instance variables */
  CheckboxGroup     queryType;
  Checkbox          textQuery, linkQuery, repSites;
  private TextField tfQuery;

  CheckboxGroup     engineOptions;
  Checkbox          allEngines, selectEngines;
  List              engines;

  Checkbox          fileSelect;
  private TextField tfOutputFile;

  private TextField tfExpandFactor, tfRootSetSize, tfNumResults;
  private TextField tfMaxLinkWeight, tfQueryRefineFactor;

  private TextField tfNSites, tfPhase, tfLeft;
  private TextField tfServQIn, tfServQOut, tfServQUtil;
  private boolean   vis;
  private Button    queryRefine, addSites;
  private Button    searchLearn,searchOnly;

  /* public methods */

  /**  */
  public AgentGUI(Agent  agent,
               Trace  trace )
  {
    this.agent = agent;
    this.trace = trace;

    int   gridy = 0, gridx = 1;
    Font  fntTitle = new Font("Helvetica",  Font.ITALIC, 18);
    Font  fnt      = new Font("TimesRoman", Font.PLAIN,  12);
    Font  fntBold  = new Font("TimesRoman", Font.BOLD,   12);

    Panel              p      = new Panel();
    ScrollPane         scroll = new ScrollPane();
    GridBagLayout      gbl    = new GridBagLayout();
    GridBagConstraints c      = new GridBagConstraints();

    vis = true;
```

```
p.setFont  (fnt);
p.setLayout(gbl);

// prepare the panel's title
/*
GuiAddFields.AddLabel ("Knowledge Agent Window",
                      true,
                      0, gridy++, 4, 1,
                      GridBagConstraints.CENTER,
                fntTitle,
                      gbl,   p          );
GuiAddFields.AddEmptyLine (gridy++, fnt, gbl, p);
GuiAddFields.AddEmptyLine (gridy++, fnt, gbl, p);
*/
//*************************************************************

GuiAddFields.AddLabel
                  ("Enter Query/Add Repository Sites",
                   false,
                   0, gridy++, 2, 1,
                   GridBagConstraints.CENTER,
               fntBold,
                   gbl,  p                    );

// add the radio buttons and the file dialog
queryType = new CheckboxGroup();
textQuery = new Checkbox ("Text query",       queryType, true);
linkQuery = new Checkbox ("Sample URL query", queryType, false);
repSites  = new Checkbox ("Repository Sites", queryType, false);

GuiAddFields.AddCheckbox(textQuery, 0, gridy, 1,1,gbl,p,(Frame)this);
tfQuery = GuiAddFields.AddTextField
                      (45,
                       1, gridy++, 4, 1,
                       GridBagConstraints.WEST,
                       false,
                       gbl,  p        );
tfQuery.setEditable(true);
GuiAddFields.AddCheckbox(linkQuery, 0, gridy, 1,1,gbl,p,(Frame)this);
GuiAddFields.AddCheckbox(repSites,  0, gridy+1, 1,1,gbl,p,(Frame)this);

GuiAddFields.AddLabel ("     Query Refinement Factor:",
                       false,
                       1, gridy, 2, 1,
                       GridBagConstraints.CENTER,
                       fnt,
                       gbl,  p        );
tfQueryRefineFactor= GuiAddFields.AddTextField
                      (5,
                       3, gridy, 1, 1,
                       GridBagConstraints.WEST,
                       false,
                       gbl,  p        );
tfQueryRefineFactor.setText("2");
queryRefine = GuiAddFields.AddButton ("Refine Query", 4, gridy++,
                             GridBagConstraints.CENTER,
                             gbl, p, (Frame)this);
addSites    = GuiAddFields.AddButton ("Add Sites", 1, gridy++,
                             GridBagConstraints.WEST,
                             gbl, p, (Frame)this);
```

```
        GuiAddFields.AddEmptyLine (gridy++, fnt, gbl, p);
        //**********************************************************
        GuiAddFields.AddLabel
                          ("Choose Search Engines:",
                           false,
                           0, gridy, 2, 1,
                           GridBagConstraints.CENTER,
                       fntBold,
                           gbl,   p                   );
        GuiAddFields.AddLabel
                          ("Various Arguments:",
                           false,
                           3, gridy++, 2, 1,
                           GridBagConstraints.CENTER,
                       fntBold,
                           gbl,   p                   );

        // add the radio buttons and the scrollable list
        engineOptions = new CheckboxGroup();
        allEngines    = new Checkbox ("All Engines", engineOptions, false);
        selectEngines = new Checkbox ("Select From List -->  ",
                              engineOptions, true    );

        GuiAddFields.AddCheckbox(allEngines,
                           0, gridy, 1,1,gbl,p,(Frame)this);
        GuiAddFields.AddCheckbox(selectEngines,
                           0, gridy+1, 1,1,gbl,p,(Frame)this);

        engines = new List(SearchEngine.NUM_ENGINES,true);
        for ( int i=0; i<SearchEngine.NUM_ENGINES; i++ )
          engines.addItem(SearchEngine.engineNames[i]);
        GuiAddFields.AddComponent (engines, 1, gridy,1,2,gbl,p);

        GuiAddFields.AddLabel ("Max. Link Weight:",
                              false,
                              3, gridy, 2, 1,
                              GridBagConstraints.CENTER,
                              fnt,
                              gbl,   p        );
        tfMaxLinkWeight     = GuiAddFields.AddTextField
                             (5,
                              5, gridy, 1, 1,
                              GridBagConstraints.WEST,
                              false,
                              gbl,   p        );
        tfMaxLinkWeight.setText("1.25");

        gridy+=2;
        GuiAddFields.AddEmptyLine (gridy++, fnt, gbl, p);

        //**********************************************************

        GuiAddFields.AddLabel
                          ("Set Size Restrictions",
                           false,
                           0, gridy++, 2, 1,
                           GridBagConstraints.CENTER,
                       fntBold,
                           gbl,   p                   );
        GuiAddFields.AddLabel ("Root Set Size:",
                               false,
                               0, gridy, 1, 1,
                               GridBagConstraints.CENTER,
                               fnt,
                               gbl,  p        );
        tfRootSetSize = GuiAddFields.AddTextField
                              (3,
                               1, gridy, 1, 1,
                               GridBagConstraints.WEST,
                               false,
                               gbl,  p        );
        tfRootSetSize.setText("30");

        GuiAddFields.AddLabel ("Expand Factor:",
                               false,
                               2, gridy, 1, 1,
                               GridBagConstraints.CENTER,
                               fnt,
                               gbl,  p        );
        tfExpandFactor = GuiAddFields.AddTextField
                              (3,
                               3, gridy, 1, 1,
                               GridBagConstraints.WEST,
                               false,
                               gbl,  p        );
        tfExpandFactor.setText("40");

        GuiAddFields.AddLabel ("Show Best:",
                               false,
                               4, gridy, 1, 1,
                               GridBagConstraints.CENTER,
                               fnt,
                               gbl,  p        );
        tfNumResults = GuiAddFields.AddTextField
                              (3,
                               5, gridy++, 1, 1,
                               GridBagConstraints.WEST,
                               false,
                               gbl,  p        );
        tfNumResults.setText("15");

        GuiAddFields.AddEmptyLine (gridy++, fnt, gbl, p);

        //**********************************************************

        GuiAddFields.AddLabel
                          ("Execution Trace",
                           false,
                           0, gridy++, 1, 1,
                           GridBagConstraints.CENTER,
                      fntBold,
                           gbl,  p                    );

        GuiAddFields.AddLabel ("Number Of Sites:",
                               false,
                               0, gridy, 1, 1,
                               GridBagConstraints.CENTER,
                               fnt,
                               gbl,  p        );
        tfNSites = GuiAddFields.AddTextField
```

```
                        (5,
                         1, gridy, 1, 1,
                         GridBagConstraints.WEST,
                         false,
                         gbl,  p        );
tfNSites.setText("0");
tfNSites.setEditable(false);

GuiAddFields.AddLabel ("URLs Left:",
                         false,
                         2, gridy, 1, 1,
                         GridBagConstraints.CENTER,
                         fnt,
                         gbl,  p        );
tfLeft = GuiAddFields.AddTextField
                        (4,
                         3, gridy, 1, 1,
                         GridBagConstraints.CENTER,
                         false,
                         gbl,  p        );
tfLeft.setText("");
tfLeft.setEditable(false);

GuiAddFields.AddLabel
                     ("Dedicated Rdrs:",
                      false,
                      4, gridy, 1, 1,
                      GridBagConstraints.CENTER,
                 fnt,
                      gbl,  p                  );
tfServQUtil= GuiAddFields.AddTextField
                     (3,
                 5, gridy++, 1, 1,
                 GridBagConstraints.WEST,
                 false,
                 gbl,  p        );
tfServQUtil.setText("");
tfServQUtil.setEditable(false);

GuiAddFields.AddLabel ("Phase:",
                         false,
                         0, gridy, 1, 1,
                         GridBagConstraints.CENTER,
                         fnt,
                         gbl,  p        );
tfPhase = GuiAddFields.AddTextField
                        (35,
                         1, gridy, 3, 1,
                         GridBagConstraints.WEST,
                         false,
                         gbl,  p        );
tfPhase.setText("");
tfPhase.setEditable(false);

tfServQIn  = GuiAddFields.AddTextField
                     (3,
                 4, gridy, 1, 1,
                 GridBagConstraints.WEST,
                 false,
                 gbl,  p        );
tfServQIn.setText("");
```

```
        tfServQIn.setEditable(false);
        tfServQOut = GuiAddFields.AddTextField
                          (3,
                       5, gridy++, 1, 1,
                       GridBagConstraints.WEST,
                       false,
                       gbl,  p        );
        tfServQOut.setText("");
        tfServQOut.setEditable(false);
        GuiAddFields.AddLabel
                          ("Pending Read Reqs/Ans",
                           false,
                           4, gridy++, 2, 1,
                           GridBagConstraints.CENTER,
                       fnt,
                           gbl,  p                   );

        //**********************************************************

        GuiAddFields.AddLabel
                          ("Choose Output File:",
                           false,
                           0, gridy++, 1, 1,
                           GridBagConstraints.CENTER,
                       fntBold,
                           gbl,  p                   );
        GuiAddFields.AddButton ("Select File...", 0, gridy,
                          GridBagConstraints.WEST,
                                gbl, p, (Frame)this);

        tfOutputFile = GuiAddFields.AddTextField
                              (45,
                               1, gridy, 4, 1,
                               GridBagConstraints.WEST,
                               false,
                               gbl,  p        );
        tfOutputFile.setEditable(true);

        GuiAddFields.AddEmptyLine (gridy++, fnt, gbl, p);

        //**********************************************************

        GuiAddFields.AddButton ("Trace...", 0, gridy, GridBagConstraints.WEST,
                                gbl, p, (Frame)this);
        GuiAddFields.AddEmptyLine (gridy++, fnt, gbl, p);

        //**********************************************************

        GuiAddFields.AddButton ("Save State", 0, gridy,
                          GridBagConstraints.CENTER,
                                gbl, p, (Frame)this);
        GuiAddFields.AddButton ("Restore State", 1, gridy,
                          GridBagConstraints.CENTER,
                                gbl, p, (Frame)this);
        searchLearn =
        GuiAddFields.AddButton ("Search & Learn", 3, gridy,
                          GridBagConstraints.CENTER,
                                gbl, p, (Frame)this);
        searchOnly =
        GuiAddFields.AddButton ("Search Only", 5, gridy,
                          GridBagConstraints.CENTER,
                            gbl, p, (Frame)this);

    gridy+=2;
    GuiAddFields.AddEmptyLine (gridy++, fnt, gbl, p);
    //**********************************************************

    GuiAddFields.AddButton ("Exit Agent", 2, gridy,
                      GridBagConstraints.CENTER,
                            gbl, p, (Frame)this);

    //**********************************************************

    /* general settings */
    setBackground(Color.gray);
    addWindowListener(new DWAdapter());

    scroll.add(p);
    add("Center", scroll);
    add("West", new Label("  "));
    add("East", new Label("  "));
    add("South", new Label("  "));
  }

  public void itemStateChanged(ItemEvent   event)
  {
    if (event.getItemSelectable() == allEngines)
    {
      engines.setVisible(false);
    }
    else if (event.getItemSelectable() == selectEngines)
    {
      engines.setVisible(true);
    }
    else if (event.getItemSelectable() == textQuery)
    {
      tfQuery.setEditable(true);
      tfRootSetSize.setVisible(true);
      tfQueryRefineFactor.setVisible(true);
      queryRefine.setVisible(true);
      searchLearn.setVisible(true);
      searchOnly.setVisible(true);
      addSites.setVisible(false);
    }
    else if (event.getItemSelectable() == linkQuery)
    {
      tfQuery.setEditable(true);
      tfRootSetSize.setVisible(false);
      tfQueryRefineFactor.setVisible(false);
      queryRefine.setVisible(false);
      searchLearn.setVisible(true);
      searchOnly.setVisible(true);
      addSites.setVisible(false);
    }
    else if (event.getItemSelectable() == repSites)
    {
      tfQuery.setEditable(true);
      tfQueryRefineFactor.setVisible(false);
      queryRefine.setVisible(false);
      searchLearn.setVisible(false);
      searchOnly.setVisible(false);
      addSites.setVisible(true);
      }
    }

    public void actionPerformed(ActionEvent event)
    {
      String command = event.getActionCommand();

      if ( command == "Refine Query" )
      {
        Agent.Action action = (Agent.Action) new Agent.ActionRefineQuery
                      (tfQuery.getText(),
                   new Integer (tfQueryRefineFactor.getText()).intValue(),
                   tfQuery);
        agent.setAction(action);
      }
      else
      if ( command == "Add Sites" )
      {
        Agent.Action action = (Agent.Action) new Agent.ActionAddSites
                      (tfQuery.getText());
        agent.setAction(action);
      }
      else
      if (command == "Select File..." )
      {
        FileDialog fd = new FileDialog(this, "Agent File");
        fd.show();
        if (fd.getDirectory() != null && fd.getFile() != null )
        tfOutputFile.setText(fd.getDirectory()+fd.getFile());
      }
      else
      if (command == "Trace..." )
      {
        TraceDialog td = new TraceDialog(this,
                                 "Agent Trace Options",
                                 Trace.AGENT_TRACE,
                                 trace             );
        td.show();
      }
      else
      if (command == "Save State")
      {
        FileDialog fd = new FileDialog(this, "Agent File");
        fd.show();
        if (fd.getDirectory() != null && fd.getFile() != null )
        {
        String fileName = fd.getDirectory()+fd.getFile();
        try
        {
          ObjectOutputStream oos =
            new ObjectOutputStream(new FileOutputStream(fileName));
          agent.saveState(oos);
          oos.flush();
          oos.close();
        }
        catch (Exception e)
        {
          System.err.println("Error while saving the Agent's state:\n"+e);
        }
        }
        }
        else
        if (command == "Restore State")
        {
          FileDialog fd = new FileDialog(this, "Agent File");
          fd.show();
          if (fd.getDirectory() != null && fd.getFile() != null )
          {
          String fileName = fd.getDirectory()+fd.getFile();
          try
          {
            ObjectInputStream ois =
              new ObjectInputStream(new FileInputStream(fileName));
            agent.restoreState(ois);
            ois.close();
          }
          catch (Exception e)
          {
            System.err.println("Error while saving the Agent's state:\n"+e);
          }
          }
        }
        else
        if ( command == "Exit Agent")
        {
          Agent.Action action = new Agent.ActionExit();
          agent.setAction(action);
        }
        else
        if (command == "Search & Learn" || command == "Search Only")
        {
          boolean update = command == "Search & Learn" ? true : false;
          boolean         goodInput = true;
          SearchEngine[]  searchEngines = null;

          if ( tfOutputFile.getText().length() == 0 )
          {
          StatusDialog sd = new StatusDialog
            (this,"Bad Input","Output File Must Be Chosen!");
          sd.show();
          goodInput = false;
          }
          if ( textQuery.getState()             &&
             tfRootSetSize.getText().length() == 0 )
          {
          StatusDialog sd = new StatusDialog
            (this,"Bad Input","Size of Root Set Must Be Specified!");
          sd.show();
          goodInput = false;
          }
          if ( tfQuery.getText().length() == 0 )
          {
          StatusDialog sd = new StatusDialog
            (this,"Bad Input","Query Must Be Specified!");
          sd.show();
          goodInput = false;
          }
          if ( tfExpandFactor.getText().length()  == 0 ||
             tfNumResults.getText().length()    == 0    )
          {
          StatusDialog sd = new StatusDialog
```

```
            (this,"Bad Input","Expand Factor/Num. Results Must Be Set!");
          sd.show();
          goodInput = false;
          }
          if ( allEngines.getState() )
          {
          searchEngines = new SearchEngine [SearchEngine.NUM_ENGINES];
          try {
            // prepare an array of all search engines
            for ( int i = 0; i < SearchEngine.NUM_ENGINES; i++ )
              searchEngines[i] = SEManager.ENGINES[i].seClone();
          }
          catch (CloneNotSupportedException e)
            { /* clone is supported by the Search Engine classes*/}
          }
          else
          {
          int[] selectedEngines = engines.getSelectedIndexes();

          if ( selectedEngines.length == 0 )
            {
            StatusDialog sd = new StatusDialog
              (this,"Bad Input","Engines Must Be Chosen!");
            sd.show();
            goodInput = false;
          }
          else
            {
            searchEngines = new SearchEngine[selectedEngines.length];
            try
            {
              // prepare an array with clones of the selected engines
              for ( int i=0; i<selectedEngines.length; i++ )
                searchEngines[i]= (SearchEngine)
                SEManager.ENGINES[selectedEngines[i]].seClone();
            }
            catch (CloneNotSupportedException e)
            { /* clone is supported by the Search Engine classes*/ }
          }
          }

          if (goodInput)
          {
          Agent.Action action = textQuery.getState() ?
            (Agent.Action) new Agent.ActionTextQuery
                       (tfQuery.getText(),
                      // Aya 20.10.99 - changed the paradigm, the user must refine
  the query
                      //                explicityly by clicking the refine query
  button
                      //              no automatic refinement during query execution
                      /*                new Integer
  (tfQueryRefineFactor.getText()).intValue(), */
                      0,
                      new Integer (tfRootSetSize.getText()).intValue(),
                      new Integer (tfExpandFactor.getText()).intValue(),
                      new Integer (tfNumResults.getText()).intValue(),
                      new Float   (tfMaxLinkWeight.getText()).floatValue(),
                      tfOutputFile.getText(),
                      searchEngines,
                      update) :
```

```
          (Agent.Action) new Agent.ActionLinkQuery
                    (tfQuery.getText(),
                   new Integer (tfExpandFactor.getText()).intValue(),
                   new Integer (tfNumResults.getText()).intValue(),
                   new Float   (tfMaxLinkWeight.getText()).floatValue(),
                   tfOutputFile.getText(),
                   searchEngines,
                   update);
        agent.setAction(action);
        }
      }
      else
      {
        System.out.println(event);
      }

      return;
    }

    public final void
    setPhase (String phase)
    {
      tfPhase.setText(phase);
    }
    public final void
    setNSites (int n)
    {
      tfNSites.setText(String.valueOf(n));
    }
    public final void
    setLeft (int n)
    {
      tfLeft.setText(String.valueOf(n));
    }
    public final void
    setServQIn (int n)
    {
      tfServQIn.setText(String.valueOf(n));
    }
    public final void
    setServQOut (int n)
    {
      tfServQOut.setText(String.valueOf(n));
    }
    public final void
    setServQUtil (int n)
    {
      tfServQUtil.setText(String.valueOf(n));
    }
    class DWAdapter extends WindowAdapter
    {
      public void windowClosing(WindowEvent event)
      {
        agent.exit();
      }
    }

  }
```